Cryptographic Combinatorial Clock-Proxy Auctions

Cryptographic Combinatorial Clock-Proxy Auctions

David C. Parkes[1], Michael O. Rabin[1], and Christopher A. Thorpe[1]

School of Engineering and Applied Sciences
Harvard University
parkes@eecs.harvard.edu, rabin@seas.harvard.edu, cat@eecs.harvard.edu

Abstract. We present a cryptographic protocol for conducting efficient, provably-correct and secrecy-preserving combinatorial clock-proxy auctions. The "clock phase" functions as a trusted auction despite price discovery: bidders submit encrypted bids, and prove for themselves that they meet activity rules, and can compute total demand and thus verify price increases without revealing any information about individual demands. In the sealed-bid "proxy phase", all bids are revealed the auctioneer via time-lapse cryptography and a branch-and-bound algorithm is used to solve the winner-determination problem. Homomorphic encryption is used to prove the correctness of the solution, and establishes the correctness of the solution to any interested party. Still an NP-hard optimization problem, the use of homomorphic enryption imposes additional computational time on winner-determination that is linear in the size of the branch-and-bound search tree, and thus roughly linear in the original (search-based) computational time. The result is a solution that avoids, in the usual case, the exponential complexity of previous cryptographically-secure combinatorial auctions.

1 Introduction

While there now exist practical protocols for cryptographic auctions of identical items, and practical methods of computing optimal outcomes in non-cryptographic combinatorial auctions, we know of no practical protocol for conducting a *cryptographic combinatorial auction*, in which a seller offers various quantities of distinct goods, buyers bid on bundles of these goods, and cryptography provides both secrecy and provable correctness. By secrecy, we mean that the auctioneer cannot exploit bid information to change the outcome of the auction, and by provable correctness, we mean that the auctioneer is obligated to issue proofs of correctness to prove he did not deviate from the posted auction rules.

Indeed, the optimization problem associated with winner determination for combinatorial auctions is NP-hard and computing the outcome of such an auction in a secure manner is therefore a significant challenge. We describe a cryptographic auction protocol that meets our secrecy and provable-correctness requirements, elicits accurate bids, and achieves a significant efficiency improvement over earlier solutions. Whereas all previous methods incur exponential computational cost, our solution avoids exponential cost in the usual case because we

can emply the use of branch-and-bound search, with additional cyrptographic proof work that scales linearly in the size of the branch-and-bound search tree. Indded, one important contribution is to develop a *general framework for proving the correctness of a solution to mathematical optimization problems*, where the input and constraints are encrypted.

The particular combinatorial auction that we study is the *combinatorial clock-proxy auction* (CCP) [1], which is a simple and efficient protocol for conducting combinatorial auctions. It was originally developed for auctions of wireless spectrum but is applicable in many other domains such as those of airport landing slots and power generation rights. This auction combines a simple price discovery ("clock") phase with a sealed-bid round ("proxy") phase.[1]

In the clock phase, the auctioneer creates a "clock" for each item for sale that represents the current price at which that item is to be sold, starting with low prices and increasing the price across rounds. In a sequence of rounds, bidders submit a bundle of the items they desire at the current clock prices. Whenever the demand exceeds the supply for a good, the clock price increases for that good in the next round. The clock phase ends when there is no excess demand for any good. At this point, bidders can submit additional bids, which, together with the clock bids, form the bids that define the input to the proxy phase. The proxy phase (or simply "proxy auction") is a second price, sealed-bid auction.

In our *cryptographic combinatorial clock-proxy* (CCCP) auction, all bid information is encrypted, and these encryptions are posted to the public. No party, including the auctioneer, can decrypt any values until all bids have been submitted in both phases. After all bids are in, only the auctioneer receives the decryption key, computes the outcome in private, reveals individual outcomes to each bidder, and issues efficiently-checkable proofs that the reported outcomes are correct given the public encrypted bids. This complete secrecy until the auction closes removes opportunities for collusion while assuring that the process remains trusted and verifiable by all participants, offering an unprecedented balance of efficiency, privacy, and transparency.

In non-cryptographic auctions, trust can be made possible at the cost of privacy via disclosure. Indeed, this is one path that Ausubel et al. [1], the designers of CCP suggest. But this can be undesirable for a number of reasons: bidders may not want competitors to learn about the values of their bids even after the fact; it may be politically undesirable to reveal that the winning bidder was willing to pay much more that was charged via the auction rules, and revealing bids received during the clock phase may lead to new opportunities for collusion.[2] Ausubel et al. [1] also argue that the confidentiality of values is of primary importance in an implementation, and suggest that in some areas of

[1] Porter et al.[2] earlier described a combinatorial-clock auction, and Parkes and Ungar [3] and Ausubel and Milgrom [4] described variants on the proxy auction phase.

[2] In a recent FCC auction for the 700MHz spectrum the government has for the first time removed all feedback about the particular bids submitted in each round. Each bidder receives individualized feedback about its own bid activity. Clearly this higher degree of secrecy brings along the need for increased trust in the auctioneer.

the auction, some values should be hidden even from the auctioneer: "Only the computer need know." Our techniques complement such a "black box" system by guaranteeing the results are correct, not simply that the programs on the system are believed to be correct.

We advance several technical contributions in the present work. During the clock phase, we employ homomorphic cryptography to protect the secrecy of bids while allowing bidders to prove they satisfy "activity rules" and allowing everyone to compute the aggregate demand for goods that determines the next round's prices. As in our previous work on non-combinatorial sealed bid auctions [5], we employ time-lapse cryptography [6], to provide secrecy during the bidding process while enforcing *nonrepudiation*: guaranteed revelation of the bids to the auctioneer when the bidding is complete. This avoids protocol completion incentive problems [7] in which bidders who realizing they will lose or change their minds can refuse to complete a distributed commercial protocol.

In the primary technical contribution, *we demonstrate how to use our cryptographic framework to prove the correctness of solutions to general classes of integer linear optimization problems*; this is how we efficiently compute the auction outcome and prove it correct. Our auctioneer employs branch-and-bound, mixed-integer programming search techniques to compute the outcome in private, avoiding costly secure computation for the optimization task; he can then prove that the outcome is correct with efficiently-checkable proofs. This seems to us to open up the possibility, for the first time, of large-scale, provably-correct combinatorial auctions.

1.1 Related work

A body of existing research considers the use of cryptographic methods to provide trust without compromising privacy; see Brandt [8] and Parkes et al. [5] for a recent discussion. Much of the previous work focuses on non-combinatorial sealed bid auctions with *complete privacy*, where no party learns anything except the outcome [9–12]. We previously advanced the security model adopted here, that of an auctioneer who must prove every action correct, and who learns bid information only after the auction closes—preventing meaningful disclosures [5].

We are only aware of one collection of research, by Yokoo and Suzuki [13], that considers cryptographic combinatorial auctions in depth. While their pioneering work offers a theoretical solution to an important problem, their solutions, which require exponential computations to prove the auction correct, can scale only to very small auctions in practice. One method they provide is based on dynamic programming using polynomial secret sharing to compute the optimal solution to the combinatorial optimization problem without revealing the inputs. Another method employs homomorphic encryption [14], but again fails to scale because computation is performed explicitly on each of the exponentially many possible allocations of goods. The same authors also extend their work to remove the need for a third-party auctioneer [15], but are again limited by the scalability of dynamic programming in this domain and also by additional process complexity implied by such a completely distributed solution. Naor et al. [11] have also proposed the use of garbled circuits to compute the outcome of a combinatorial

auction. Though the work is important for its foresight and theoretical affirmative results, we know of no practical implementation of obfuscated circuits that has been applied to significant real-world problems on the scale of a commercial combinatorial auction.

2 Cryptographic preliminaries

Several cryptographic systems support the secrecy-preserving, provably-correct computation that we employ to conduct the auction. Because Paillier's cryptosystem [16] supports all of the operations we need and is widely accepted in secure protocols, we use it in our exposition. That said, there is nothing that necessitates the use of Paillier's system; in fact, other solutions can be constructed that are computationally more efficient, but may complicate the protocol. These include, among others, Pedersen commitments [17] and ElGamal encryption [18], based on the hardness of computing discrete logarithms modulo a prime, and the provably correct secure computation system described by Rabin et al. [19].[3] We reserve for future work a complete discussion of how these and other systems might also support our protocol.

Due to special mathematical properties Paillier encryption enjoys, it is possible for a *Prover* (in our application the Auctioneer) to create a random permutation S' of a set of encryptions S so that a verifier believes that S' encrypts precisely the same set of values that S does. In the spirit of our work, this can be done in a manner not revealing any information about the encrypted values.

In the Paillier cryptosystem, one can generate a new "random-looking" encryption of a particular element by multiplying it by a encryption of 0 — we call this a "re-encryption factor". The auctioneer can create many random permutations of the encrypted values and commit to the re-encryption factors in each permutation. The *Verifier* then asks the auctioneer to reveal the re-encryption factors for some of the permutations, and verifies that the factors are well-formed (that is, they are encryptions of zero) and that the permutation is correct. The remaining permutations, for which the factors remain unrevealed, are now verified correct with high probability. Cryptographers have formalized this idea as a "shuffle", or "mix network".[4][5]

We will employ a mix network to create a verifiable random permutation of the encrypted bids that are submitted to the proxy auction. This will allow

[3] We have devised a similar protocol to the one we describe based on Pedersen commitments; while this protocol is computationally more efficient, it is mathematically more sophisticated, and we present the Paillier-based solution here because of the simplicity that a protocol with a single cryptosystem enjoys.

[4] The latter term should not be confused with hard-to-trace network communications protocols that are sometimes referred to by the same name.

[5] See Abe et al. [20, 21] for early work on such permutation networks, and Boneh and Golle [22] for an excellent formalization of mix networks, a brief survey of other solutions, and an interesting efficient protocol for proving a mix network is correct with high (but not overwhelming) probability. Boneh and Golle's efficient solution should not be employed without using an additional mechanism to verify its correctness. See Boneh and Golle [22].

the branching decisions of the branch-and-bound proof tree to be published without revealing any information about the actual underlying inputs to the linear optimization problems; bidders can thereby be satisfied with their outcome without learning private bid information.

3 Combinatorial auctions

We consider a multi-unit combinatorial allocation problem with goods $G = \{G_1, \ldots, G_m\}$ and bidders $B = \{B_1, \ldots, B_n\}$. There are C_j units of each good G_j available and each bidder B_i has a valuation function $v_i(s_i)$ on bundles $s_i \in \mathbb{Z}^m_{\geq 0}$, where $s_{ij} \leq C_j$ denotes the number of units of item G_j in the bundle.

An *efficient allocation* solves $V^* = \max_{s \in \mathbb{F}} \sum_i v_i(s_i)$ where $\mathbb{F} = \{s : \sum_i s_{ij} \leq C_j,\ \forall j \in G\}$ and $s = (s_1, \ldots, s_n)$ denotes the allocation of items to bidders.

We assume quasi-linear utility u_i (or *payoff* π_i), so that bidder B_i's utility for bundle s_i, given payment $y_i \in \mathbb{R}_{\geq 0}$, is $\pi_i = u_i(s_i, y_i) = v_i(s_i) - y_i$. We make the standard assumptions of *normalization*, with $v_i(s_i) = 0$ when $s_{ij} = 0$ for all items G_j, and *free disposal*, with $v_i(s_i) \geq v_i(s'_i)$ for $s'_i \geq s_i$.

Bidder	Bid	Items	Price
1	1	$\{A, B\}$	3
2	1	$\{B, C\}$	3
3	1	$\{A, C, D\}$	3
4	1	$\{C, D, E\}$	2
5	1	$\{E, F\}$	4.5
6	1	$\{F\}$	3
7	1	$\{D\}$	1

Table 1. A simple example of a combinatorial auction problem

An example of a combinatorial auction problem is illustrated in Table 1. This example has 7 bids each from a unique bidder, and 6 goods $G = \{A, B, \ldots, F\}$, all in unit supply. In this case each bidder is single-minded, and only interested in a single bundle of goods. The example is adapted from Sandholm et al. [23]. In the efficient allocation, the winners are bidders {1,5,7}, for a total value of 8.5. (By tie-breaking, another outcome that is just as good is to select $\{2, 5, 8\}$ as winners.)

The payments in the proxy auction are selected from the *buyer-optimal core*. Consider the payoff vector $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ induced by an efficient allocation s^* and payment vector $y = \langle y_1, \ldots, y_n \rangle$, i.e. with $\pi_i = v_i(s^*_i) - y_i$. Let π_0 denote the payoff to the seller, which is the total revenue received by the seller, i.e. $\pi_0 = \sum_i y_i = V^* - \sum_i \pi_i$. A payoff profile $\langle \pi_0, \pi \rangle$ is in the *core* if $\pi_0 + \sum_{i \in K} \pi_i \geq V(K)$ for all $K \subseteq B$, where $V(K) = \max_{s \in \mathbb{F}} \sum_{k \in K} v_k(s_k)$. This states that no coalition of $K \subseteq B$ buyers and the seller can improve its total payoff by leaving the auction and allocating the items amongst itself, leaving all members weakly better off. Simple algebra shows that the core payoffs can be equivalently defined as:

$$\mathit{Core} = \{\pi : \sum_{i \in W \setminus K} \pi_i \leq V^* - V(K),\ \forall K \subseteq W, \pi_i \geq 0, \pi_i \leq v_i(s^*_i)\},$$

where W is the set of *winners* in the efficient allocation s^*. The *buyer-optimal core* defines a payoff vector that solves $\overline{\pi} \in \arg\max_{\pi \in Core} \sum_i \pi_i$.

The buyer-optimal core is related to the outcome of the Vickrey-Clarke-Groves (VCG) mechanism [24]. The VCG mechanism defines payments so that the payoff to bidder i is $\pi_i^{\text{vcg}} = V^* - V(B \setminus \{i\})$, i.e., each bidder's payoff is the marginal value it contributes by its presence. In general, $\sum_i \overline{\pi}_i < \sum_i \pi_i^{\text{vcg}}$ and the revenue to the seller is greater in a buyer-optimal core outcome than in the VCG mechanism. But when the VCG outcome is in the core then it coincides with the (unique) buyer-optimal core outcome. In the general case, the buyer-optimal core is not unique and the final payments in the proxy auction are selected to minimize the maximal difference to the VCG payoff across all buyer-optimal core outcomes.[6]

In the example in Table 1, the payoff to winning buyers $\{1, 5\}$ and 7 in the VCG mechanism is $8.5 - 8.5 = 0, 8.5 - 8 = 0.5$ and $8.5 - 8 = 0.5$ respectively, with corresponding payments $\{\$3, \$4\}$ and $\$0.5$. It is easily checked that this outcome is in the core, and thus also the buyer-optimal core outcome.

4 Phase One: The Clock Auction

The presentation of our main results begins by considering the first phase of the CCP auction, which is the *clock-auction* phase. The clock phase proceeds in rounds until demand is weakly less than supply for every good. In each round t, a price vector $p^t = \langle p_1^t, \ldots, p_m^t \rangle$ associates prices with each good: p_j^t is the price for good G_j in round t. The price vector is initialized to low prices (although not necessarily uniformly across all goods) for the first round, $t = 1$, and is increased in each successive round based on the amount of excess demand. Bidders submit a bid $s_i^t \in \mathbb{Z}_{\geq 0}^m$ in each round. These bids are ultimately included within the proxy bids that form the input to the proxy phase.

We are interested in supporting this price discovery process, but without allowing any party—the auctioneer included—to learn anything about any bids not already implied by the public information. Following the description of Ausubel et al. [1], we allow the price increase on a good in a round to depend on the amount of excess demand on that good.[7] One requirement, then, is that any party (the auctioneer included) must be able to determine the excess demand on each good in the current round without learning anything else about the current bids. It will also be necessary to allow any party to verify that the bids meet an activity rule that restricts bidding strategies, in particular a *revealed preference activity rule*, and without revealing any information.

All bids made during the clock phase must also be submitted as proxy bids in the proxy phase. We ensure this and prevent non-repudiation through the use

[6] This particular choice follows the suggestion of *threshold payments* in Parkes et al. [25] in the context of a combinatorial exchange, and as refined in the context of the proxy auction by Day and Raghavan [26].

[7] Ausubel et al. [1] also discuss the idea of using *intra-round bids* in which the auction proceeds in a smaller number of discrete rounds and bidders express quantity demands in each round at all prices along a price trajectory that will be traced during the round. We save this extension for future work.

of a time-lapse cryptography (TLC) service [6]. At the start of the auction, the auctioneer in CCCP announces the initial price vector p^1 and the supply $C = \langle C_1, \ldots, C_m \rangle$ and designates a *public time-lapse cryptographic key* N. Because the secret key corresponding to N (and based on the factorization of N) is not revealed until after all bidder information has been submitted, the auctioneer cannot reveal private information that could affect the outcome. The forced reconstruction of N guarantees that the bids can be opened by the auctioneer when the auction is complete.[8]

At the beginning of round t, the auctioneer publishes the current clock price vector $p^t = \langle p_1^t, \ldots, p_m^t \rangle$. Then, each bidder B_i publishes an encrypted version of her bid given the current prices: $E(s_i^t) = \langle E(s_{i1}^t, r_{i1}^t), \ldots, E(s_{im}^t, r_{im}^t) \rangle$. Bidders publish these encrypted bundles to all bidders, the auctioneer and any verifiers, either by broadcast or to a common "bulletin board" during a fixed period of time for round t. This encrypted bundle is represented as a vector of length m, in which each coefficient s_{ij}^t is an encryption of the quantity B_i wants for good G_j at price p_j^t. The values r_{ij}^t are independent, fresh *random help values* that each bidder selects in accordance with the probabilistic homomorphic encryption scheme, and kept secret. Encryptions of zero must be included for any undesired item to keep the number of items in the bundle secret.

Bid Validity and Activity Rules Each bidder must now prove that the bid is valid and satisfies an *activity rule*.[9] The basic idea in a revealed-preference activity rule (RPAR) is to require bidders to follow a demand-revealing strategy that is consistent with some fixed valuation function across all clock rounds. Consider a current round t and some previous round $t' < t$, corresponding price vectors p^t and $p^{t'}$, and B_i's associated demands s_i^t and $s_i^{t'}$. A straightforward bidder with valuation v_i prefers s_i^t to $s_i^{t'}$ when prices are p^t, i.e. $v_i(s_i^t) - p^t \cdot s_i^t \geq v_i(s_i^{t'}) - p^t \cdot s_i^{t'}$, and prefers $s_i^{t'}$ to s_i^t when prices are $p^{t'}$, i.e. $v_i(s_i^{t'}) - p^{t'} \cdot s_i^{t'} \geq v_i(s_i^t) - p^{t'} \cdot s_i^{t'}$. Adding these two inequalities (the values of the bundles cancel) yields the activity rule, i.e. $(p^t - p^{t'}) \cdot (s_i^t - s_i^{t'}) \leq 0$.

Before proving the RPAR, bidders must prove that their current demands are valid by using an interval proof: each B_i proves for the demand for good G_j, $0 \leq s_{ij}^t \leq C_j$. That is, the demand lies in the interval between 0 and the auction's capacity for that good.[10]

---

[8] The TLC service in Rabin et al. [6] creates a time-lock ElGamal key, but it can also create any cryptographic key for which a verifiable distributed key generation protocol exists, including Paillier keys (like RSA keys, the product of two large primes).

[9] While we talk about the "bidder" proving various facts about the bid history to the auctioneer and any other interested party, we of course intend the proofs to be generated by a computer program running on secure hardware controlled by the bidder, both to maintain the security of any private information and because the cryptographic computations should not be carried out by hand.

[10] We also require that the capacities C_j are less than half the modulus of the cryptosystem ($N/2$), but as the moduli are typically hundreds or thousands of bits, this poses no practical problems.

Each bidder can now readily prove that she satisfies the activity rule using homomorphic cryptography via the clock prices and the published encrypted bids. This must be established in round t with respect to all previous rounds $t' < t$. The details of this are presented in Appendix A.1.

Computing Aggregate Demand At the conclusion of each round, the aggregate demand for each item must be computed. The aggregate demand vector s^t for all goods at the end of round t is simply:

$$s^t = \langle \sum_{i=1}^{n} s_{i1}^t, \ldots, \sum_{i=1}^{n} s_{im}^t \rangle$$

Given the encrypted demand vectors, we can compute use the homomorphic properties of the cryptosystem to compute an encryption of the aggregate demand vector s^t as follows:

$$E(s^t) = \langle \prod_{i=1}^{n} E(s_{i1}^t, r_{i1}^t), \ldots, \prod_{i=1}^{n} E(s_{im}^t, r_{im}^t) \rangle \quad (1)$$

$$= \langle E(\sum_{i=1}^{n} s_{i1}^t, \prod_{i=1}^{n} r_{i1}^t), \ldots, E(\sum_{i=1}^{n} s_{im}^t, \prod_{i=1}^{n} r_{im}^t) \rangle \quad (2)$$

By multiplying each bidder's encrypted demand for an item together, we obtain an encryption of the sum of all bidders' demands for that item; the random help value of this encryption is the product of the random help values from all bidders' encrypted demands. Since the secret decryption key does not yet exist, decryption can only be performed by unlocking the encrypted value with its random help value.

While the random help value could be directly constructed from the other values, such a direct computation would reveal too much, because each encrypted demand's random help value would unlock that particular demand. We thus employ another well-known cryptographic protocol, a simple, secure multi-party computation of a product of secret values, to compute the random help values needed to unlock the aggregate demand. We sketch the protocol but omit a more detailed description for reasons of space.

After each round t, we repeat the following process for each good G_j, obtaining the above aggregate demand vector (Eq. 2). B_i constructs shares of the random help value associated with the demand for good G_j, so that the product of these shares equals the random help value r_{ij}^t. B_i then distributes these shares among all bidders. Once all the shares are received, the bidders multiply their received shares together, yielding random factors of the help value $\prod i = 1^n r_{ij}^t$. Then, bidders broadcast these random factors to all bidders, and multiply them together to yield the desired help value. This allows anyone to decrypt the encrypted sum of the aggregate demand for that good and verify the result. Recall that since the encrypted individual demands are public, one can compute an encryption of their sum by multiplying the encryptions.

We remark without proof that this sub-protocol to compute the random help values is information-theoretically secure and reveals no information other than

the results. Furthermore, it requires only two broadcasts and scales linearly in the number of items for sale. Moreover, bidders who refuse to participate in this protocol to compute the aggregate demand can be disqualified, and the demand recomputed without them. If a bidder submits incorrect values during this protocol, then the computed values r_j^t will be discovered to be incorrect.[11]

4.1 Transition to the Proxy Phase

Let T denote the number of rounds in the clock phase. Each bidder has submitted a bid on $\langle s_i^1, \ldots, s_i^T \rangle$ bundles at public prices $\langle p^1, \ldots, p^T \rangle$. A bidder can now:

(a) improve any bid submitted during the clock phase

(b) include bids on additional bundles

These additional bids are committed by each bidder, by encrypting with the key associated with the TLC service and then sharing them, for instance posting them to a public bulletin board. When the auctioneer receives the time-lapse decryption key he will then prove that each bidder meets the activity rules that constrain her ability to bid in this transition from clock to proxy.

For (a), we first require each bidder B_i to associate a bid price $b_i(s_i^t)$ with every bid. This bid price must satisfy:

$$b_i(s_i^t) \geq p^t \cdot s_i^t \tag{3}$$

For (b), each bidder can also submit additional bids, which we index $k > t$ to indicate that they are received after the close of the clock phase. Consider some bundle s_i^k, either one of the clock bundles or one of these additional bundles, and its associated bid price $b_i(s_i^k)$. Any such bid must satisfy the following constraints:

$$b_i(s_i^k) - p^t \cdot s_i^k \leq \alpha(b_i(s_i^t) - p^t \cdot s_i^t), \quad \forall t \in \{1, \ldots, T\} \tag{4}$$

This requires that the bidder would not have been much happier (by some relaxation parameter $\alpha \geq 1$) by bidding this bundle in any clock round than the bundle that it did bid in that round. We will also require each bidder to *pad* her bids (with zero bids), so that the total number of bundles that receive a bid is constant across all bidders. Let K denote the number of such bids.

Once this transition round closes the auctioneer receives the time-lapse decryption key and will now generate a proof that all bids satisfy these activity rules (Eq. 3 and 4).[12] If a bidder submits a non-compliant bid at this phase, the auctioneer can prove the bid is non-compliant and remove any such bids from the computation of the outcome.

---

[11] Although the process we describe cannot detect *which* bidder submitted incorrect values, the auctioneer can resort to a more sophisticated *verifiable* secret sharing protocol (e.g., [27]) that can identify non-compliant bidders. The ability to use such protocols if necessary should discourage malicious bidders from attempting to disrupt our protocol: they can always be discovered and disqualified. The auctioneer can also recover and reveal disqualified bidders' prior bids once he receives the time-lapse decryption key.

[12] To establish the activity rule, then for every bidder B_i and round $t \in \{1, \ldots, T\}$, the auctioneer computes provably correct encryptions of the dot products $p^t \cdot s_i^t$

5 Phase Two: The Proxy Auction

The proxy phase of the CCP auction is used to determine the final allocation of goods and the final payments. This requires solving a sequence of optimization problems. Given that the winner-determination problem for combinatorial auctions is NP-hard, it seems to us *essential* that the bids are now revealed to the auctioneer in plain text. This enables the auctioneer to leverage efficient, branch-and-bound methods of integer programming in determining the outcome. We reiterate that the auctioneer is unable to submit or alter bids, or change the outcome of the auction in any way, once the bids are revealed. And up until this point neither the auctioneer or any other party has received any information about the bids.

The main technical innovation is to use cryptographic methods to prove that a solution to an integer program is optimal by establishing various linear constraints implied by a "fathomed" (or solved) *branch-and-bound* tree. An appealing aspect of our approach is that it is *completely agnostic to the particular heuristics by which a branch-and-bound proof tree is generated (e.g. depth-first, breadth-first, memory management, branch-selection heuristics, etc.).* Rather, the system works directly with the information that is established upon the conclusion of the search, i.e. from the final proof tree.

We confine our solution to what can be considered a standard, textbook treatment of branch-and-bound search (e.g., see Wolsey [28]). In doing so, we impose two main restrictions on the use of branch-and-bound algorithms: (a) no pre-processing, and (b) no cut-generation. While modern optimization solvers, such as ILOG's CPLEX, do make extensive use of both of these methods, good performance can be achieved on reasonably sized problems without either feature. Nevertheless, this presents a very appealing avenue for future work.

5.1 Branch-and-Bound Search

To illustrate the principle of branch-and-bound search we will consider the winner-determination problem (WDP) in the proxy phase. In defining this, we index the proxy bids $s_i = \langle s_{i1}, \ldots, s_{iK} \rangle$ from each bidder i. Recall that K is the total number of bids received from each bidder (by padding if necessary.) Let $b_i = \langle b_{i1}, \ldots, b_{iK} \rangle$ denote the associated bid values. The integer programming (IP) formulation for the WDP is:

$$\max \left\{ \sum_i \sum_k x_{ik} b_{ik} \ : \ \text{s.t. } x \in \mathbb{F}, x_{ik} \in \{0,1\}, \ \forall i, \forall k \right\} \tag{5}$$

for values bid during the clock phase. He further computes, for every bidder B_i, the $t(K-T)$ dot products $p^t \cdot s_i^k, \forall t \in \{1, \ldots, T\} \forall k \in \{T+1, \ldots, K\}$. These dot products are computed in the same way encrypted dot products are computed at the end of Section 4. To prove Eq. 4, he shows that the bidder prefers each final proxy bid $\langle s_i^k, b_i(s_i^k) \rangle, T < k \leq K$, he computes the encrypted differences of these encrypted dot products and encrypted bid values $b_i(s_i^k)$ and $b_i(s_i^t)$ (respectively) and multiplies the second result by the public constant α; this allows him to use a simple interval proof to demonstrate the inequality.

where

$$\mathbb{F} = \left\{ \begin{array}{ll} \sum_i \sum_k s_{ikj} x_{ik} & \leq C_j, \ \forall j \in G, \\ \sum_k x_{ik} & \leq 1, \ \forall i \in B \end{array} \right\}, \tag{6}$$

and these constraints ensure that no more units of a good are allocated than in the supply and that no more than one bid is accepted from any single bidder.

In describing branch-and-bound, let $\underline{z}$ denote the value of the best solution found so far (initialized to $-\infty$), and let $\underline{x}$ denote that solution (undefined when no solution has been found.) This is the *incumbent* solution. The first step in branch-and-bound is to solve the linear-programming (LP) relaxation,

$$\max \left\{ \sum_i \sum_k x_{ik} b_{ik} \ : \ \text{s.t. } x \in \mathbb{F}, x_{ik} \geq 0, \ \forall i, \forall k \right\} \tag{7}$$

Let $L^0 = \{x : x \in \mathbb{F}, \ x_{ik} \geq 0, \forall i, \forall k\}$ denote the LP-relaxation of the solution space. Let $\overline{x}^0$ denote the solution on L^0 and $\overline{z}^0$ the value of this solution. If $\overline{x}^0$ is *integral* then branch-and-bound can stop with $\underline{x} := \overline{x}^0$ and $\underline{z} := \overline{z}^0$. The solution $\overline{x}^0$ will in general be *fractional*, meaning that one or more of the variables has a value that is neither 0 or 1.

To illustrate this, consider again the example in Table 1 and let x_{i1} denote the variable corresponding to the bid from each agent i. In the example, the solution to the LP relaxation is fractional, with an assignment $\langle 0.5, 0.5, 0.5, 0, 1, 0, 0.5 \rangle$ and total value of 9.5. When this occurs, a branching decision is made on one of the fractional variables. Continuing with the example, suppose that we branch on $x_{71} \leq 0$ and $x_{71} \geq 1$. This generates two new sub-problems, one defined on solution space $L^1 = \{x : x \in \mathbb{F}, x_{71} \leq 0, \ x_{ik} \geq 0, \ \forall i, \forall k\}$ and one defined on solution space $L^2 = \{x : x \in \mathbb{F}, x_{71} \geq 1, \ x_{ik} \geq 0, \ \forall i, \forall k\}$. Branch-and-bound continues by picking one of these and solving the associated linear program.

Let $(L^p, \overline{x}^p, \overline{z}^p)$ denote the associated LP and solution. In any one of the following three cases, this becomes a *"fathomed"* (or solved) leaf:

(a) the subproblem is infeasible

(b) the subproblem has an integral optimal solution; if $\underline{z} < \overline{z}^p$ then $\underline{z} := \overline{z}^p$ and $\underline{x} := \overline{x}^p$.

(c) the subproblem is feasible and the solution fractional, but $\beta \overline{z}^p \leq \underline{z}$ for some $\beta \leq 1$ that controls the optimality tolerance.

In our example, the solution to L^2 is integral and we would set $\underline{z} := \overline{z}^2 = 8.5$ and $\underline{x} := \overline{x}^2 = \langle 1, 0, 0, 0, 1, 0, 1 \rangle$. This leaf is now fathomed. But the solution to L^1 is fractional ($\overline{x}^1 = \langle 0.5, 0.5, 0.5, 0, 1, 0, 0 \rangle$) and has value $\overline{z}^1 = 9 \not\leq \underline{z} = 8.5$. In such a case, branch-and-bound search will generate two additional subproblems, typically by doing something like branching on the most fractional variable. The unsolved subproblems are stored on the "open list." Branch-and-bound finally terminates when the open list is empty, returning the incumbent as the solution. Finishing with the example, when we branch on $x_{11} \leq 0$ and $x_{11} \geq 1$ we obtain two leaves that are fathomed. The LP relaxations generate integral solutions and their value is less than that of the solution already found.

While there are many sophisticated strategies for managing the details of a branch-and-bound search, for our purposes all that is required is a *fathomed* branch-and-bound tree, i.e. one for which all leaves have been fathomed. An example of a so-called *proof tree* for the example is shown in Figure 5.2.

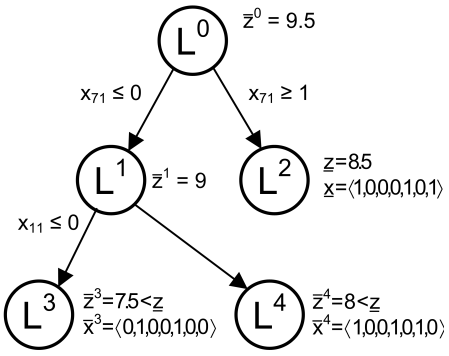

Fig. 1. Branch-and-Bound Proof Tree

5.2 Establishing Correctness of Integer Program Solutions

In this section we describe the general approach to establish the correctness of the solution to an integer program (IP). Along the way we also provide a method to establish the correctness of the solution to a linear program (LP). Recall that the *input* to the IP is published in encrypted form. In describing our approach we assume that the solution to the IP is revealed to all parties, but this is not necessary. All relevant steps can instead be performed using an encryption of the solution, if the solution itself is to remain private.

The cryptographic proof is constructed from a proof tree, as generated at the termination of a branch-and-bound search. To perform these steps on the encrypted inputs, we first note that IPs, LPs and their duals are all defined with linear inequalities and linear objective functions. Therefore, we can prove that a set of constraints are satisfied, or that a solution has a particular objective value, using the verifiable addition, subtraction and multiplication operations, and equality and inequality tests, on Paillier-encrypted values. All that is required are encryptions of all the private inputs (the bids in our case).

Because we have formulated all inputs as integers, it is theoretically possible to obtain LPs with rational coefficients at every point in the proof tree, which implies that they have rational solutions. Moreover, since any computation on the rationals can be performed by an equivalent computation on the integers (with at most a constant factor increase in the number of steps), we can employ established cryptographic techniques that prove integer computations correct for rational numbers as well. This allows us to calculate and prove correct exact solutions to rational LPs.[13]

[13] In practice, it is likely that the results will be computed using a computer program that yields a floating-point or real number as a result. We can instead convert this

The proof of the correctness of a solution x^* to a IP proceeds with the following steps:

1. Any permutation-invariance in the class of problems being solved is leveraged for the purpose of secrecy by generating a random permutation using a mix network as described in Section 2. This proves to verifiers that the set of encrypted values in the proof tree is the same as the set of inputs, but makes the correspondence between those sets is unknown.[14]
2. The branching decisions that define the proof tree are revealed. (For instance, "at the root the left branch is $x_6 \leq 0$ and the right branch is $x_6 \geq 1$" and so on.) The amount of information that this reveals depends on the amount of permutation invariance in the class of problems. For example, if all inputs can be "mixed" with all other inputs then this reveals no information.
3. The solution x^* to the IP is revealed along with a claim $\beta \leq 1$ about its optimality (e.g., $\beta = 9999/10000$ would state that the solution quality is within multiplicative factor 9999/10000 of the optimal solution.) The encrypted solution $E(x^*)$ is published and shown to be a valid encryption of x^*: this is because many of our operations only apply to two encrypted operands, and for those we need to use $E(x^*)$ rather than the unencrypted x^*.
4. Let q^* denote the leaf associated with the optimal solution. This is revealed by the prover. The prover then proceeds to:
   (a) Publish $E(V^*)$ and prove that its value is correct (i.e. the value is an encryption of the objective value of the IP given solution x^*).
   (b) Prove that x^* satisfies the constraints of the LP formulated at leaf L^{q^*} (i.e. prove inequalities defined in terms of the encrypted input to the IP and also the additional inequalities implied by the branching decisions.)
   (c) Prove that x^* is integral.
5. Consider every leaf q (including the optimal leaf) in turn. For every such leaf, the prover then proceeds to:
   (a) Let y^q denote the solution to the dual LP at leaf L^q and D^q the value of that dual solution. Publish the encrypted dual $E(y^q)$ solution and the encrypted dual value $E(D^q)$ at this leaf.
   (b) Prove that the dual solution satisfies the constraints of the dual LP formulated at leaf L^q.
   (c) Prove the correctness of the dual value $E(D^q)$ by reference to the dual formulation, and that $\beta E(D^q) \leq E(V^*)$.

This procedure encompasses both leaves that are fathomed by infeasibility and leaves that are fathomed by bound in the same way. Note that a leaf that is

---

value to a rational number and prove that the constraints are satisfied with acceptably small error.

[14] A complete permutation invariance is not required for this step. For example, in the context of the combinatorial auction application, we seek a permutation of the order of the bids submitted by a particular bidder and also a permutation across bidders. But we should not mix-up bids submitted by one bidder with bids submitted by another bidder.

infeasible in its primal form has a dual solution with value $-\infty$ by the duality theory of LP. Therefore, the prover can always construct a feasible dual solution to prove that there is no better (primal) solution in the feasible solution space that corresponds to a particular leaf. It should be easy to see how to generalize the above approach to a mixed integer program.[15]

5.3 Application: Winner Determination

We now instantiate the general approach to the WDP for combinatorial auctions. Recall that (s_{ik}, b_{ik}) denotes the kth proxy bid submitted by bidder i, where bundle s_{ik} contains s_{ikj} units of item $j \in G$. The IP formulation for the WDP is:

$$\max_{x_{ik}} \sum_{i \in B} \sum_{k} x_{ik} b_{ik} \qquad \text{WDP(B)}$$

$$\text{s.t.} \quad \sum_{i \in B} \sum_{k} s_{ikj}\ x_{ik} \leq C_j, \quad \forall j \in G \qquad (8)$$

$$\sum_{k} x_{ik} \leq 1, \quad \forall i \in B \qquad (9)$$

$$x_{ik} \in \{0, 1\}, \quad \forall i \in B, \forall k$$

where x_{ik} indicates whether the kth bid from bidder i is accepted. We label this formulation WDP(B) to make explicit that this is problem is defined for all bidders and to allow for variations WDP(L) defined on a subset $L \subseteq B$ of bidders. Constraints (8) ensure that the supply constraints are satisfied. Constraints (9) ensure that no bidder receives more than one bundle of items.[16]

Once the solution x^* is published and associated with a leaf of the branch-and-bound tree, and once it has been shown to satisfy the constraints of the appropriate restricted-primal formulation for the leaf (se the Appendix) and also to be integral, the remaining work in proving the optimality is in terms of establishing properties for the dual of this restricted primal formulation for each leaf of the search tree. All the information required to complete these proofs is either available in the encrypted proxy bids (e.g. s_{ikj}, b_{ik}), publicly known (e.g. the capacity C_j), or defined by the branching decisions that are published by the mechanism.

5.4 Determining Payments and Announcing Results

The final step in the CCP auction is to find the buyer-optimal core point that minimizes the maximal deviation across all buyers from the payoff profile in

[15] In the case that the original problem is an LP rather than a IP then there is no proof tree to deal with, and the procedure simplifies to: (a) publish $E(V^*)$ and prove this value is correct; (b) prove that x^* satisfies the constraints of the LP; (c) publish an encrypted dual solution $E(y^q)$ and associated dual value $E(D^q)$; (d) prove that the solution is dual feasible, and that $\beta E(D^q) \leq E(V^*)$.

[16] Details about the linear-programming relaxation of WDP(B) and the corresponding dual DWDP(B), along with the restricted primal and dual formulations for the leaf of a winner-determination branch-and-bound tree are provided in Appendix A.2.

the VCG mechanism, as discussed in Section 3. The details of this step are provided in Appendix A.3, and require solving and proving the correctness of sequence of optimization problems (each of which is a simple variant on the winner determination problem), and ultimately establishing the correctness of a solution to a linear program to determine the final payments.

Taken together, the above steps are sufficient to prove to any interested party that the allocation and payments are correct. But because we employed a mix network to prevent bidders from learning the position of their bids in the proof tree, we still need to convince an individual bidder that the particular allocation announced for them is correct for them. This is easy to achieve by privately revealing to each bidder *only* the correspondence between their original proxy bid that was accepted and its position in the permutation generated by the mix network. The bidder will then be satisfied that the outcome proven is correct from her perspective because she can verify that her bid was allocated in the optimal allocation. She will similarly believe that the payment corresponding to the bidder that submitted the bid, and hence her own payment, is correct.[17]

6 Conclusions

We have described a cryptographic method to enable secret and provably-correct, combinatorial auctions. Whereas previous methods incur exponential cost in providing a secure solution to the NP-hard winner-determination problem, we can use branch-and-bound algorithms and generate a proof with overhead that is linear in the size of the ultimate branch-and-bound tree, and thus linear in the computational search time. In doing so, the solution presented here will avoid exponential time complexity with overwhelming probability. Our particular focus has been on the practically important combinatorial clock-proxy auction, which is used by governments in high-stakes settings. It bears additional emphasis that in striving for what we consider to be a practical solution, we require that the auctioneer is trusted not to reveal information about bids once an auction has closed. This is the same tradeoff that we made in our earlier work on non-combinatorial auctions [5]. In making this tradeoff, we achieve a system that is provably correct and trustworthy, and we believe can be implemented in a realistic business setting on cost-effective computing hardware.

[17] This does imply that a small amount of information that is leaked by our system, over and above that implied by the outcome of the auction: each bidder learns where in the various proof trees her own accepted bid was branched on. But this appears to us to disclose no useful information to a bidder.

References

1. Ausubel, L., Cramton, P., Milgrom, P.: The clock-proxy auction: A practical combinatorial auction design. In Cramton, P., Shoham, Y., Steinberg, R., eds.: Combinatorial Auctions. MIT Press (January 2006)
2. Porter, R., Shoham, Y.: On cheating in sealed bid auctions. In: Proc. ACM Conf. on Electronic Commerce (EC'03). (2003)
3. Parkes, D.C., Ungar, L.H.: Iterative combinatorial auctions: Theory and practice. In: Proc. 17th Nat. Conf. on Artificial Intelligence (AAAI-00). (2000) 74–81
4. Ausubel, L.M., Milgrom, P.: Ascending auctions with package bidding. Frontiers of Theoretical Economics **1** (2002) 1–42
5. Parkes, D.C., Rabin, M.O., Shieber, S.M., Thorpe, C.A.: Practical secrecy-preserving, verifiably correct and trustworthy auctions. Electronic Commerce Research and Applications **7** (2008) 294–312
6. Rabin, M.O., Thorpe, C.: Time-lapse cryptography. Technical Report TR-22-06, Harvard University School of Engineering and Computer Science (2006)
7. Bradford, P.G., Park, S., Rothkopf, M.H.: Protocol completion incentive problems in cryptographic Vickrey auctions. In: 7th Int. Conference on Electronic Comemrce Research (ICECR-7). (2004) 55–64
8. Brandt, F.: How to obtain full privacy in auctions. International Journal of Information Security (2006) 201–216
9. Franklin, M.K., Reiter, M.K.: The design and implementation of a secure auction server. IEEE Transactions on Software Engineering **22**(5) (1996) 302–312
10. Harkavy, M., Tygar, J.D., Kikuchi, H.: Electronic auctions with private bids. In: Proc. 3rd USENIX Workshop on Electronic Commerce. (1998)
11. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proc. First ACM Conf. on Elec. Commerce. (1999) 129–139
12. Lipmaa, H., Asokan, N., Niemi, V.: Secure Vickrey auctions without threshold trust. In: Proc. 6th International Conference on Financial Cryptography (FC 2002). (2002) 87–101
13. Suzuki, K., Yokoo, M.: Secure combinatorial auctions by dynamic programming with polynomial secret sharing. In: Sixth International Financial Cryptography Conference (FC-2002). (2002) 44–56
14. Suzuki, K., Yokoo, M.: Secure generalized Vickrey auction using homomorphic encryption. In: Seventh International Financial Cryptography Conference (FC-2003). (2003) 239–249
15. Yokoo, M., Sakurai, Y., Matsubara, S.: The effect of false-name bids in combinatorial auctions: New Fraud in Internet Auctions. Games and Economic Behavior **46**(1) (2004) 174–188
16. Paillier, P.: Public-key cryptosystems based on composite residuosity classes. In: Proc. EUROCRYPT '99. (1999) 223–239
17. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Advances in Cryptology - CRYPTO. Lecture Notes in Computer Science, Springer Verlag (1991) 129–140
18. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Information Theory **IT-31**(4) (1985) 469–472
19. Rabin, M.O., Servedio, R.A., Thorpe, C.: Highly efficient secrecy-preserving proofs of correctness of computations and applications. In: Proc. IEEE Symposium on Logic in Computer Science. (2007)

20. Abe, M.: Mix-networks on permutation networks. In: Proc. ASIACRYPT '99. Volume 1716 of Lecture Notes in Computer Science., Springer (1999) 258–273
21. Abe, M., Hoshino, F.: Remarks on mix-network based on permutation networks. In: Public Key Cryptography. Volume 1992 of Lecture Notes in Computer Science., Springer (2001) 317–324
22. Boneh, D., Golle, P.: Almost entirely correct mixing with applications to voting. In: CCS '02: Proceedings of the 9th ACM conference on Computer and communications security, New York, NY, USA, ACM (2002) 68–77
23. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. Management Science **51**(3) (2005) 374–390
24. Nisan, N.: Introduction to mechanism design (for computer scientists). Cambridge University Press (2007)
25. Parkes, D.C., Kalagnanam, J.R., Eso, M.: Achieving budget-balance with Vickrey-based payment schemes in exchanges. In: Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01). (2001) 1161–1168
26. Day, R.W., Raghavan, S.: Fair payments for efficient allocations in public sector combinatorial auctions. Management Science (2006)
27. Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient multiparty computations secure against an adaptive adversary. Lecture Notes in Computer Science **1592** (1999) 311 ff.
28. Wolsey, L.A.: Integer Programming. John Wiley (1998)

A Appendix: Further Details of the Auction Protocol

A.1 Establishing the Activity Rule: First, since the price vectors $p^{t'}$ and p^t are public, anyone can compute the price difference vector $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_m \rangle = p^t - p^{t'}$. Second, using the encrypted demand vectors $E(s_i^t)$ and $E(s_i^{t'})$, the homomorphic properties of the cryptosystem allow computing B_i's encrypted demand difference vector $\hat{s}_i = \langle \hat{s}_{i1}, \ldots, \hat{s}_{im} \rangle = s_i^t - s_i^{t'}$:

$$
\begin{aligned}
E(s_i^t) &= \langle E(s_{i1}^t, r_{i1}^t), \ldots, E(s_{im}^t, r_{im}^t) \rangle \\
E(s_i^{t'}) &= \langle E(s_{i1}^{t'}, r_{i1}^{t'}), \ldots, E(s_{im}^{t'}, r_{im}^{t'}) \rangle \\
E(\hat{s}_i) &= \langle \frac{E(s_{i1}^t, r_{i1}^t)}{E(s_{i1}^{t'}, r_{i1}^{t'})}, \ldots, \frac{E(s_{im}^t, r_{im}^t)}{E(s_{im}^{t'}, r_{im}^{t'})} \rangle \\
&= \langle E(s_{i1}^t - s_{i1}^{t'}, r_{i1}^t / r_{i1}^{t'}), \ldots, E(s_{im}^t - s_{im}^{t'}, r_{im}^t / r_{im}^{t'}) \rangle
\end{aligned}
$$

To compute the encrypted dot product of the price difference vector and the encrypted demand difference vector, $E(\hat{p} \cdot \hat{s}_i)$, we can again use the homomorphic properties of the cryptosystem:

$$
\begin{aligned}
E(\hat{p} \cdot \hat{s}_i) &= E(\hat{s}_{i1}, r_{i1}^t / r_{i1'}^t)^{\hat{p}_1} \times \ldots \times E(\hat{s}_{im}, r_{im}^t / r_{im}^{t'})^{\hat{p}_m} \\
&= E(\hat{p}_1 \times \hat{s}_{i1}, r_{i1}^t / r_{i1}^{t'}) \times \ldots \times E(\hat{p}_m \times \hat{s}_{im}, r_{im}^t / r_{im}^{t'}) \\
&= E(\hat{p}_1 \times \hat{s}_{i1} + \ldots + \hat{p}_m \times \hat{s}_{im}, r_{i1}^t / r_{i1}^{t'} \times \ldots \times r_{im}^t / r_{im}^{t'})
\end{aligned}
$$

We adopt $\hat{r}_i$ to notate the random help value encrypting the dot product (the last formula above): $\hat{r}_i = r^t_{i1}/r^{t'}_{i1} \times \ldots \times r^t_{im}/r^{t'}_{im}$. We now have an encryption of this dot product—a single value that proves the activity rule when it is less than or equal to zero.[18] Consequently, each bidder now proves using another interval proof that this encrypted value is less than (but relatively close to) zero. Our example shows that B_i can compute the precise random help value corresponding to the encryption of a dot product of an encrypted vector with a public vector. This allows B_i to prove facts about the result like any other value it encrypted and even though the decryption key has not yet been constructed.

A.2 Detailing the LP Relaxations for Winner Determination: The linear-programming relaxation of WDP(B) is defined by replacing $x_{ik} \in \{0, 1\}$ with $x_{ik} \geq 0$. In defining the dual (and overloading notation from the clock phase, which is no longer needed), we introduce variables p_j to denote the dual variable for constraints (8) and π_i to denote the dual variable for constraints (9). Given this, then the dual problem is:

$$\min_{p,\pi} \sum_j C_j p_j + \sum_i \pi_i \qquad \text{DWDP(B)}$$

$$\text{s.t.} \quad \sum_j s_{ikj}\ p_j + \pi_i \geq b_{ik}, \quad \forall i, k \qquad (10)$$

$$p_j \geq 0, \pi_i \geq 0$$

A sequence of branching decisions leading to a fathomed leaf in the search tree introduces additional constraints to WDP(B) and modifies the dual problem at the leaf. Let $(i, k) \in OUT$ indicate that branch $x_{ik} \leq 0$ has been taken and $(i, k) \in IN$ denote that branch $x_{ik} \geq 1$ has been taken. Given these constraints, the restricted primal and dual pair becomes:

$$\max_{x_{ik}} \sum_i \sum_k x_{ik} b_{ik} \qquad \text{RWDP(B)}$$

$$\text{s.t.} \quad \sum_i \sum_k s_{ikj}\ x_{ik} \leq C_j, \quad \forall j \in G \qquad (11)$$

$$\sum_k x_{ik} \leq 1, \quad \forall i \qquad (12)$$

$$x_{ik} \leq 0, \quad \forall (i, k) \in OUT \qquad (13)$$

$$x_{ik} \geq 1, \quad \forall (i, k) \in IN \qquad (14)$$

$$x_{ik} \geq 0, \quad \forall i, \forall k$$

[18] If the bidder does not prove the activity rule, then the bid is invalid and the auction rules should dictate whether the bidder must resubmit, or be disqualified for the round.

$$\min_{p,\pi,\delta} \sum_j C_j p_j + \sum_i \pi_i - \sum_{i|(i,k)\in W} \delta_i \qquad \text{DRWDP(B)}$$

$$\text{s.t.} \quad \sum_j s_{ikj}\ p_j + \pi_i \geq b_{ik}, \quad \forall (i,k) \notin (OUT \cup IN) \tag{15}$$

$$\sum_j s_{ikj}\ p_j + \pi_i - \delta_i \geq b_{ik}, \quad \forall (i,k) \in IN \tag{16}$$

$$p_j \geq 0, \pi_i \geq 0, \delta_i \geq 0$$

Dual variable δ_i corresponds to constraints (14) in RWDP(B). The variable that dualizes constraints (13) drops out of the dual formulation because it appears with coefficient zero in the objective and appears in a non-binding constraint.

A.3 Determining the Proxy Payments: To determine the payments we must determine the payoffs in the buyer-optimal core that minimize the maximal deviation across all buyers from the VCG payoff profile. Solving for this point requires the use of constraint generation, but the cryptographic proof can be constructed after-the-fact in terms of just the final set of constraints. By a slight reformulation of the method in Day and Raghavan [26], the payoffs to winning buyers $i \in W$ can be computed in the following LP:

$$\max_{\pi,m} \sum_{i\in W} \pi_i - \epsilon\ m \qquad \text{EBOP}$$

$$\text{s.t.} \quad \sum_{i\in W\setminus L} \pi_i \leq V^* - V(L), \quad \forall L \subseteq W \tag{17}$$

$$\pi_i + m \geq \pi_i^{\text{vcg}}, \quad \forall i \in W \tag{18}$$

$$0 \leq \pi_i, \quad \forall i \in W$$

$$0 \leq m,$$

with $\pi_i = 0$ for all $i \notin W$, and for some small $\epsilon > 0$. The objective is to maximize the total buyer payoff, but then for small ϵ to break ties in favor of minimizing the maximal deviation m from the VCG payoffs across all such buyers. Constraints (17) are the core constraints and constraints (18) force m to adopt the maximal difference to VCG payoffs. Given a solution π^* to EBOP, the payments collected from each winning buyer $i \in W$ are $b_i(s_i^*) - \pi_i^*$.

EBOP is an LP and has no integer variables. But notice that part of its input has required solving IPs (since constraints (17) are defined in terms of V^* and $V(L)$). More difficult, there are an exponential number of constraints (17). Day and Raghavan [26] suggest using *constraint generation* to construct a subset $\mathcal{L} \subseteq 2^W$ of coalitions, with constraints (17) reformulated as $\sum_{i\in W\setminus L} \pi_i \leq V^* - V(L), \quad \forall L \in \mathcal{L}$. Let EBOP($\mathcal{L}$) denote the relaxed form of EBOP in with just this subset of constraints. New constraints are introduced until it can be established that:

$$\max_{L \subseteq W} \sum_{i \in W \setminus L} \pi_i - (V^* - V(L)) \leq 0 \tag{19}$$

This establishes that none of the missing constraints is binding. (In practice, this is also the separation problem that is solved in generating a new constraint.) Given a solution π^* to EBOP($\mathcal{L}$), the separation problem can be formulated and solved via an IP as a simple variation on the regular WDP:

$$\max_{x_{ik}} \sum_{i \in W} \left(1 - \sum_k x_{ik}\right) \pi_i - V^* + \sum_{i \in W} \sum_k x_{ik} b_{ik} \qquad \text{SEP}(\pi^*)$$

$$\text{s.t.} \quad \sum_i \sum_k s_{ikj}\, x_{ik} \leq C_j, \quad \forall j \tag{20}$$

$$\sum_k x_{ik} \leq 1, \quad \forall i \in W \tag{21}$$

$$x_{ik} \in \{0, 1\}$$

Putting this all together, the methodology for establishing the correctness of the final payments is as follows:

1. Publish the set $\mathcal{L}$ of coalitions of winners that are used to establish the correctness of payments. (Note that this does not reveal any information if a mix network was used on the inputs.) Publish the parameter $\epsilon > 0$.
2. Publish the solution $E(\pi^*)$ and $E(m^*)$ to EBOP($\mathcal{L}$). Publish the vector of proxy payments $p^* = \langle p_1^*, \ldots, p_n^* \rangle$. Prove that $p_i^* = \sum_k x_{ik}^* b_{ik} - \pi_i^*$ for all buyers i.
3. Publish and establish the correctness of $E(\pi^{\text{vcg}})$, for $\pi^{\text{vcg}} = \langle \pi_1^{\text{vcg}}, \ldots, p_n^{\text{vcg}} \rangle$. Publish and establish the correctness of $E(V(L))$ for all $L \in \mathcal{L}$.
4. Publish and prove the solution to the separation problem SEP(π^*).
5. Prove that the solution to EBOP is primal feasible.
6. Publish an encrypted solution to the dual problem and prove it is dual feasible. Prove the value $E(D^*) \leq \beta E(V^*)$ for some parameter $\beta \geq 1$, e.g. $\beta = 100001/100000$.

Step 3 requires proving facts about solutions to different winner determination problems. For the VCG payoff, $\pi_i^{\text{vcg}} = V^* - V(B \setminus i)$ and thus this needs the value of $E(V(B \setminus i))$ to be proved correct. This can be done following the approach in the previous section for the WDP. Similarly, we need to prove the correctness of $E(V(L))$ for subsets $L \subseteq B$. Note that both kinds of proofs can be verified without revealing the solution to these subproblems, and that no useful information leaks from publishing branching decisions in the branch-and-bound search because of the use of a mix network. Step 4 can be reduced to an instance of the WDP and proved analogously. In Step 6 we need the dual to the linear program EBOP($\mathcal{L}$).